МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №5

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

Розробка ASCII ART генератора для візуалізації 3D-фігур

Виконав:

ст. гр. РІ-21сп

Павлусь В.П

Прийняв:

Шербак С.С.

Львів-2024

**Мета лабораторної роботи:** Cтворення додатка для малювання 3D-фігур у ASCII-арті на основі об’єктно-орієнтованого підходу та мови Python

**Хід роботи**

**Завдання 1: Проектування класів.**

Розробіть структуру класів для вашого генератора 3D ASCII-арту. Визначте основні компоненти, атрибути та методи, необхідні для програми.

**Завдання 2: Введення користувача.**

Створіть методи у межах класу для введення користувача та вказання 3D-фігури, яку вони хочуть намалювати, та її параметрів (наприклад, розмір, кольори).

**Завдання 3: Представлення фігури.**

Визначте структури даних у межах класу для представлення 3D-фігури. Це може включати використання списків, матриць або інших структур даних для зберігання форми фігури та її властивостей.

**Завдання 4: Проектування з 3D в 2D.**

Реалізуйте метод, який перетворює 3D-представлення фігури у 2D-представлення, придатне для ASCII-арту.

**Завдання 5: Відображення ASCII-арту.**

Напишіть метод у межах класу для відображення 2D-представлення 3D-фігури як ASCII-арту. Це може включати відображення кольорів і форми за допомогою символів ASCII.

**Завдання 6: Інтерфейс, зрозумілий для користувача.**

Створіть зручний для користувача командний рядок або графічний інтерфейс користувача (GUI) за допомогою об'єктно-орієнтованих принципів, щоб дозволити користувачам спілкуватися з програмою.

**Завдання 7: Маніпуляція фігурою.**

Реалізуйте методи для маніпулювання 3D-фігурою, такі масштабування або зміщення, щоб надавати користувачам контроль над її виглядом.

**Завдання 8: Варіанти кольорів.**

Дозвольте користувачам вибирати варіанти кольорів для їхніх 3D ASCII-арт-фігур. Реалізуйте методи для призначення кольорів різним частинам фігури.

**Завдання 9: Збереження та експорт.**

Додайте функціональність для зберігання згенерованого 3D ASCII-арту у текстовий файл

**Завдання 10: Розширені функції.**

Розгляньте можливість додавання розширених функцій, таких як тінь, освітлення та ефекти перспективи, для підвищення реалізму 3D ASCII-арту.

**Виконання роботи**

**Папка Classes:**

**Вміст файлу ascii\_art\_service.py:**

from Classes.color\_manager import ColorService

from Classes.cube import Cube

from Classes.cylinder import Cylinder

from Classes.file\_manager import FileManager

class AsciiArtService:

def \_\_init\_\_(self):

self.shape = None

self.color\_service = ColorService()

def display\_ascii\_art(self):

if not self.shape:

print("Form is not set!")

return

projection = self.shape.project\_to\_2d()

self.shape.render\_ascii(projection)

def set\_shape(self, shape\_type, size, height=None):

if shape\_type == "cube":

self.shape = Cube(size)

elif shape\_type == "cylinder":

self.shape = Cylinder(size, height or size)

else:

print("Incorrect form!")

def set\_color(self, color\_name):

color\_code = self.color\_service.get\_color\_code(color\_name)

if self.shape:

self.shape.set\_color(color\_code)

def rotate\_shape(self, angle\_x, angle\_y, angle\_z):

if self.shape:

self.shape.rotate(angle\_x, angle\_y, angle\_z)

def save\_to\_file(self, filename):

if not self.shape:

print("Form is not set!")

return

projection = self.shape.project\_to\_2d()

FileManager.save\_to\_file(filename, projection)

**Вміст файлу color\_manager.py:**

import json

class ColorService:

def \_\_init\_\_(self, color\_file="/Users/admin/Desktop/ДЗ/3 семестр/Спеціалізовані мови програмування/Lab5/config/colors.json"):

self.colors = self.load\_colors(color\_file)

def load\_colors(self, color\_file):

try:

with open('/Users/admin/Desktop/ДЗ/3 семестр/Спеціалізовані мови програмування/Lab5/Config/colors.json', 'r', encoding='utf-8') as file:

data = json.load(file)

return data["colors"]

except FileNotFoundError:

raise FileNotFoundError(f"File {color\_file} not found.")

def list\_colors(self):

return list(self.colors.keys())

def get\_color\_code(self, color\_name):

return self.colors.get(color\_name, "37")

**Вміст файлу cube.py:**

from Classes.shape3d import Shape3D

class Cube(Shape3D):

def \_\_init\_\_(self, size):

super().\_\_init\_\_(size)

def project\_to\_2d(self):

projection\_height = self.size \* 2

projection\_width = self.size \* 2

projection = [[" " for \_ in range(int(projection\_width \* 2))] for \_ in range(int(projection\_height))]

half = self.size // 2

vertices = [

(-half, -half, -half), (half, -half, -half),

(half, half, -half), (-half, half, -half),

(-half, -half, half), (half, -half, half),

(half, half, half), (-half, half, half),

]

rotated\_vertices = [self.apply\_rotation(x, y, z) for x, y, z in vertices]

edges = [

(0, 1), (1, 2), (2, 3), (3, 0),

(4, 5), (5, 6), (6, 7), (7, 4),

(0, 4), (1, 5), (2, 6), (3, 7)

]

min\_y = min(v[1] for v in rotated\_vertices)

max\_y = max(v[1] for v in rotated\_vertices)

vertical\_offset = (projection\_height // 2) - ((max\_y + min\_y) // 2)

for start, end in edges:

x1, y1, z1 = rotated\_vertices[start]

x2, y2, z2 = rotated\_vertices[end]

x1 += self.size

y1 += vertical\_offset

x2 += self.size

y2 += vertical\_offset

symbol = "." if z1 > 0 and z2 > 0 else "#"

self.draw\_line(x1 \* 2, y1, x2 \* 2, y2, projection, symbol)

return projection

**Вміст файлу cylinder.py:**

import math

from Classes.shape3d import Shape3D

class Cylinder(Shape3D):

def \_\_init\_\_(self, size, height):

super().\_\_init\_\_(size)

self.height = height

def project\_to\_2d(self):

projection\_height = self.height \* 2

projection\_width = self.size \* 2

projection = [[" " for \_ in range(int(projection\_width \* 2))] for \_ in range(int(projection\_height))]

radius = self.size // 2

points\_top = [(radius \* math.cos(theta), radius \* math.sin(theta), self.height // 2)

for theta in [i \* math.pi / 8 for i in range(16)]]

points\_bottom = [(radius \* math.cos(theta), radius \* math.sin(theta), -self.height // 2)

for theta in [i \* math.pi / 8 for i in range(16)]]

rotated\_top = [self.apply\_rotation(x, y, z) for x, y, z in points\_top]

rotated\_bottom = [self.apply\_rotation(x, y, z) for x, y, z in points\_bottom]

for i in range(len(rotated\_top)):

x1, y1, z1 = rotated\_top[i]

x2, y2, z2 = rotated\_top[(i + 1) % len(rotated\_top)]

x1 += self.size

y1 += self.size

x2 += self.size

y2 += self.size

symbol = "." if z1 > 0 and z2 > 0 else "#"

self.draw\_line(x1 \* 2, y1, x2 \* 2, y2, projection, symbol)

for i in range(len(rotated\_bottom)):

x1, y1, z1 = rotated\_bottom[i]

x2, y2, z2 = rotated\_bottom[(i + 1) % len(rotated\_bottom)]

x1 += self.size

y1 += self.size

x2 += self.size

y2 += self.size

symbol = "." if z1 > 0 and z2 > 0 else "#"

self.draw\_line(x1 \* 2, y1, x2 \* 2, y2, projection, symbol)

for top, bottom in zip(rotated\_top, rotated\_bottom):

x1, y1, z1 = top

x2, y2, z2 = bottom

x1 += self.size

y1 += self.size

x2 += self.size

y2 += self.size

symbol = "." if z1 > 0 and z2 > 0 else "#"

self.draw\_line(x1 \* 2, y1, x2 \* 2, y2, projection, symbol)

return projection

**Вміст файлу file\_manager.py:**

import os

class FileManager:

@staticmethod

def save\_to\_file(filename, projection):

os.makedirs("/Users/admin/Desktop/ДЗ/3 семестр/Спеціалізовані мови програмування/Lab5/Data", exist\_ok=True)

file\_path = os.path.join("/Users/admin/Desktop/ДЗ/3 семестр/Спеціалізовані мови програмування/Lab5/Data", filename)

try:

with open(file\_path, 'w') as file:

for row in projection:

file.write("".join(row) + "\n")

print(f"ASCII-art was saved in {file\_path}")

except Exception as e:

print(f"Error with save: {e}")

**Вміст файлу shape3d.py:**

import math

class Shape3D:

def \_\_init\_\_(self, size):

self.size = size

self.angle\_x = 0

self.angle\_y = 0

self.angle\_z = 0

self.color\_code = "37"

def set\_color(self, color\_code):

self.color\_code = color\_code

def rotate(self, angle\_x=0, angle\_y=0, angle\_z=0):

self.angle\_x += math.radians(angle\_x)

self.angle\_y += math.radians(angle\_y)

self.angle\_z += math.radians(angle\_z)

def apply\_rotation(self, x, y, z):

y, z = (y \* math.cos(self.angle\_x) - z \* math.sin(self.angle\_x),

y \* math.sin(self.angle\_x) + z \* math.cos(self.angle\_x))

x, z = (x \* math.cos(self.angle\_y) + z \* math.sin(self.angle\_y),

-x \* math.sin(self.angle\_y) + z \* math.cos(self.angle\_y))

x, y = (x \* math.cos(self.angle\_z) - y \* math.sin(self.angle\_z),

x \* math.sin(self.angle\_z) + y \* math.cos(self.angle\_z))

return int(x), int(y), int(z)

def draw\_line(self, x1, y1, x2, y2, canvas, symbol):

dx = abs(x2 - x1)

dy = abs(y2 - y1)

sx = 1 if x1 < x2 else -1

sy = 1 if y1 < y2 else -1

err = dx - dy

while True:

if 0 <= y1 < len(canvas) and 0 <= x1 < len(canvas[0]):

canvas[int(y1)][int(x1)] = symbol

if x1 == x2 and y1 == y2:

break

e2 = err \* 2

if e2 > -dy:

err -= dy

x1 += sx

if e2 < dx:

err += dx

y1 += sy

def project\_to\_2d(self):

raise NotImplementedError("Method project\_to\_2d must be implemented in a subclass!")

def render\_ascii(self, projection):

color\_code = f"\033[{self.color\_code}m"

reset\_code = "\033[0m"

for row in projection:

print(color\_code + "".join(row) + reset\_code)

**Папка Config:**

**Вміст файлу colors.json:**

{

"colors": {

"None": "0",

"Black": "30",

"Red": "31",

"Green": "32",

"Yellow": "33",

"Blue": "34",

"Fiolet": "35",

"Cyan": "36",

"Grey": "37",

"Dark-grey": "90",

"White": "97"

}

}

**Папка UI:**

**Вміст файлу user\_interface.py:**

from Classes.ascii\_art\_service import AsciiArtService

from Classes.cylinder import Cylinder

def user\_interface():

ascii\_art\_service = AsciiArtService()

while True:

ascii\_art\_service.display\_ascii\_art()

print("\n Main menu:")

print("1. Select shape, cube or cylinder.")

print("2. Select size.")

print("3. Set height, only for cylinder.")

print("4. Rotate shape.")

print("5. Save ASCII-art in file.")

print("6. Select color.")

print("7. Exit.")

choice = input("Select option: ").strip()

match choice:

case '1':

shape\_type = input("Select shape, cube or cylinder: ").strip().lower()

size = int(input("Enter size of shape: "))

height = int(input("Enter height of cylinder: ")) if shape\_type == "cylinder" else None

ascii\_art\_service.set\_shape(shape\_type, size, height)

case '2':

size = int(input("Enter new size: "))

if ascii\_art\_service.shape:

ascii\_art\_service.shape.size = size

case '3':

if isinstance(ascii\_art\_service.shape, Cylinder):

height = int(input("Enter new height of cylinder: "))

ascii\_art\_service.shape.height = height

else:

print("Changing the height is available only for the cylinder!")

case '4':

angle\_x = float(input("Enter rotation about X in degrees: "))

angle\_y = float(input("Enter rotation about Y in degrees: "))

angle\_z = float(input("Enter rotation about Z in degrees: "))

ascii\_art\_service.rotate\_shape(angle\_x, angle\_y, angle\_z)

case '5':

filename = input("Enter name file for save ASCII-art with extension .txt: ")

ascii\_art\_service.save\_to\_file(filename)

case '6':

print("Available colors:", ", ".join(ascii\_art\_service.color\_service.list\_colors()))

color\_name = input("Select color: ").strip()

ascii\_art\_service.set\_color(color\_name)

case '7':

print("Exit!")

break

case \_:

print("Incorrect option: Please, try again!")

**Вміст файлу runner.py:**

from UI.user\_interface import user\_interface

if \_\_name\_\_ == '\_\_main\_\_':

user\_interface()

Результат виконання, а саме створення та обертання фігури, зміна її кольору та збереження у файлі зображено на рис. 1, рис. 2, рис. 3 та рис. 4.

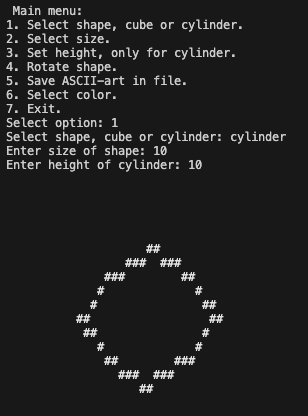


Рис. 1 – Створення циліндра



Рис. 2 – Обертання фігури

Рис. 3 – Зміна кольору фігури



Рис. 4 – Зображення фігури у файлі

**Висновок:** У ході виконання ЛР я створив високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволяє користувачам проектувати, відображати та маніпулювати 3D-фігурами в ASCII-арті, а також зберігати їх й файлах.